Cantor Meets Scott:
Semantic Foundations for
Probabilistic Networks

Steffen Smolka (Cornell)
Praveen Kumar (Cornell)
Nate Foster (Cornell & Barefoot)
Dexter Kozen (Cornell)
Alexandra Silva (UCL)
How to ensure correct behavior?

high level languages & automatic verification!

[Foster et al., ICFP 11], [Monsanto et al., POPL 12], [Kazemian et al., NSDI 12],
[Voellmy et al., SIGCOMM 13], [Khurshid et al., NSDI 13], [Nelson et al., NSDI 14],
[Anderson et al., POPL 14], [Plotkin et al., POPL 16], [Becket et al., PLDI 16],
[Subramanian et al., POPL 17], …

Assumption: network behavior is deterministic
A language for **modeling & reasoning** about networks **probabilistically**.

ProbNetKAT
A language for **modeling & reasoning** about networks **probabilistically**.

**Prob** + **NetKAT**

- probabilistic primitive: $p \oplus_r q$
- network primitives: $f:=n, \text{dup}$
A language for **modeling & reasoning** about networks **probabilistically**.

**ProbNetKAT**

- **Prob**: probabilistic primitives
  - $p \oplus_r q$
- **Net**: network primitives
  - $f:=n$, dup
- **KA**: regular expressions
  - $+, \cdot, *$
- **T**: boolean tests
  - $f=n$

- **ProbNetKAT**: 2016
- **NetKAT**: 2013
- **KAT**: 1996
- **1956**: regular expressions
- **1847**: boolean tests
A language for modeling & reasoning about networks probabilistically.

\[ \llbracket p \rrbracket \in 2^H \rightarrow \text{Dist}(2^H) \]

\[ \llbracket p \rrbracket \in 2^H \rightarrow 2^H \]

Prob + Net + KA + T

probabilistic primitives

network primitives

regular expressions

boolean tests

\[ p \oplus_r q \]

\[ f := n, \text{dup} \]

\[ +, \cdot, * \]

\[ f = n \]
Probabilistic Reasoning

ProbNetKAT model $p$, input distribution $\mu$

$\rightarrow$ traffic distribution $v = [p]^{\dagger}(\mu) \in \text{Dist}(2^H)$
Probabilistic Reasoning

ProbNetKAT model $p$, input distribution $\mu$

$\rightarrow$ traffic distribution $v = [p]^+(\mu) \in \text{Dist}(2^H)$

utilization query: $Q : 2^H \rightarrow [0,\infty]$

expected utilization: $E_v[Q]$
How to implement this?

Key Question: Approximation?
**Key Idea**

$\textbf{limits} + \text{continuity} \rightarrow \text{approximation}$

$\mu_1, \mu_2, \mu_3, \ldots \xrightarrow{\text{converges}} \mu \in \text{Dist}(2^H)$
Key Idea

limits + continuity $\rightarrow$ approximation

$\mu_1, \mu_2, \mu_3, \ldots$ converges $\rightarrow$ $\mu \in \text{Dist}(2^H)$

$f(\mu_1), f(\mu_2), f(\mu_3), \ldots$ converges $\rightarrow$ $f(\mu) \in \mathbb{R}$
Key Idea

limits + **continuity** → approximation

\[ \mu \in \text{Dist}(2^H) \]

\[ f(\mu_1), f(\mu_2), f(\mu_3), \ldots \quad \text{converges} \quad f(\mu) \in \mathbb{R} \]
Main Results

1) Iteration-free programs generate only finite distributions

2) Iteration may introduce continuous distributions … … but can be approximated by bounded iteration

3) All programs can be approximated
Main Results

1) Iteration-free programs generate only finite distributions

2) Iteration may introduce continuous distributions … … but can be approximated by bounded iteration

3) All programs can be approximated

compositionality of approximation

continuity of 

+, ·, *, ⊕
Main Results

1) Iteration-free programs generate only finite distributions

2) Iteration may introduce continuous distributions … … but can be approximated by bounded iteration

3) All programs can be approximated

4) Queries can be approximated
Main Results

1) Star-free programs generate only finite distributions.

2) Iteration may introduce continuous distributions, but can be approximated by bounded iteration.

3) All programs can be approximated.

4) Queries can be approximated.

---

**Main Results**

- Star-free programs generate only finite distributions.
- Iteration may introduce continuous distributions, but can be approximated by bounded iteration.
- All programs can be approximated.
- Queries can be approximated.

**But different topologies give different notions of limits, continuity, and approximation**
Cantor Meets Scott
## Topologies

<table>
<thead>
<tr>
<th></th>
<th>Cantor</th>
<th>Scott</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convergence</td>
<td>Weak</td>
<td>Monotone</td>
</tr>
<tr>
<td>Metric</td>
<td>✔️</td>
<td>✘</td>
</tr>
<tr>
<td>Practical Queries</td>
<td>✘</td>
<td>✔️</td>
</tr>
</tbody>
</table>

\[ d(\mu, \nu) \]

\[ \mu \sqsubseteq \nu \]
### Topologies

<table>
<thead>
<tr>
<th></th>
<th>Cantor</th>
<th>Scott</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✘</td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>

#### Example

Q : $2^H \rightarrow \mathbb{R}$

$Q(A) = |A|$

“network congestion”
# CPOs for ProbNetKAT

If a larger set of packets (in the sense of $\subseteq$) is input to the ProbNetKAT program, then the probability that a given set of packets occurs as a subset of the output set can only increase.

<table>
<thead>
<tr>
<th>History sets</th>
<th>Distributions</th>
<th>Programs</th>
<th>Query results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(2^H, \subseteq)$</td>
<td>$(D(2^H), \subseteq)$</td>
<td>$(2^H \rightarrow D(2^H), \subseteq)$</td>
<td>$(R, \leq)$</td>
</tr>
</tbody>
</table>

[Saheb-Djahromi, Jones & Plotkin]

$+, \cdot, \ast, \oplus, E[-]$ respect this order!
Summary

→ any program is approximated to arbitrary precision by finite distributions!

→ any query is approximated to arbitrary precision by finite sums!

→ convergence is monotone!

→ implemented in OCaml in ~300 LOC
Applications
Failures are a fact of life in real-world networks. Devices and links fail due to factors ranging from software and hardware bugs to interference from the environment such as loss of power or cables being severed. A recent empirical study of data center networks by Gill et al. [14] found that failures occur frequently and can cause issues ranging from degraded performance to service disruptions.

As a concrete example, consider the topology depicted in Figure 4 (a), with four switches connected in a diamond. Suppose that we wish to forward traffic from switch $S_1$ to switch $S_2$ using a scheme that encodes the local forwarding behavior of the network topology; a term using several terms: a term network—using different forwarding paths will lead to different outcomes! To cover each possible outcome, we can encode failures in ProbNetKAT using random choice and a term using several terms: a term gossip protocols.

We can encode failures in ProbNetKAT using random choice and a term using several terms: a term gossip protocols.

Obviously the answer to this question depends on the configuration of the network topology; a term using several terms: a term switches; a term

Fig. 4.

Topologies used in case studies: (a) fault tolerance, (b) load balancing, and (c) gossip protocols.

Expected number of packets traversing each link

Fault Tolerance

Utilization

Gossip protocols

Probability of delivery in the presence of failures

Expected number of nodes "infected" after $n$ rounds

$S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow S_3 \rightarrow S_1$

$S_2 \rightarrow S_1 \rightarrow S_3 \rightarrow S_4 \rightarrow S_2$

$S_3 \rightarrow S_4 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3$

$S_4 \rightarrow S_3 \rightarrow S_2 \rightarrow S_1 \rightarrow S_4$

$S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_1$

$S_2 \rightarrow S_1 \rightarrow S_4 \rightarrow S_3 \rightarrow S_2$

$S_3 \rightarrow S_4 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3$

$S_4 \rightarrow S_3 \rightarrow S_2 \rightarrow S_1 \rightarrow S_4$

$S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow S_3 \rightarrow S_1$

$S_2 \rightarrow S_1 \rightarrow S_4 \rightarrow S_3 \rightarrow S_2$

$S_3 \rightarrow S_4 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3$

$S_4 \rightarrow S_3 \rightarrow S_2 \rightarrow S_1 \rightarrow S_4$

$S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow S_3 \rightarrow S_1$

$S_2 \rightarrow S_1 \rightarrow S_4 \rightarrow S_3 \rightarrow S_2$

$S_3 \rightarrow S_4 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3$

$S_4 \rightarrow S_3 \rightarrow S_2 \rightarrow S_1 \rightarrow S_4$

$S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow S_3 \rightarrow S_1$
Topology
Routing Algorithms

ECMP, KSP, Multi, Räcke
Demand Matrix
1. Assemble ProbNetKAT Model

\[(p \cdot t)^* \cdot p\]
2. Approximate Traffic Distribution

\[ \|(p \cdot t)^* \cdot p \|_1(\mu) = \nu_1 \]
2. Approximate Traffic Distribution
2. Approximate Traffic Distribution

\[ [(p \cdot t)^* \cdot p]_3(\mu) = \nu_3 \]
2. Approximate Traffic Distribution
3. Approximate Network Metrics

\[ E_{v1}[\text{hopcount}] \leq E_{v2}[\text{hopcount}] \leq ... \]

1. \textbf{hopcount} : \text{history} \rightarrow \text{int}
2. \textbf{hopcount} \ h = \text{List.length} \ h
we briefly describe how we model the components of a network in engineering (TE) approaches have been explored. We built ProbRouting.

ProbNetKAT, extending the encodings from net2’s Abilene backbone network as shown in Figure 5. Before presenting our case studies, we can also measure expected mean latency in terms of path length:

\[
\text{Max Utilization} \quad \text{Steps of Approximation}
\]

ECMP

KSP

Multi

Räcke
Ongoing Work

Richer language (e.g. link capacities, queuing, etc.)

\[ A \rightarrow B; @1\text{Gbit/s} \]

Efficient implementation that scales to large networks

Axiomatic reasoning and a decision procedure

\[ \vdash p \equiv q \]
A continuous distribution

$(((\pi_0! \oplus \pi_1!) \cdot \text{dup})^*)$

How many paths are there? → one for every $r \in [0,1]$

What's the probability of any particular path? → 0
Taming *

Recall: $[p] \in 2^H \mapsto \text{Dist}(2^H)$

What is $[p^*]?$

$[p^*](a) := \mu_Y$

$x_0 := a$

$x_{n+1} \sim [p](x_n)$

$y := x_0 \cup x_1 \cup x_2 \cup \ldots$

Idea: stop executing loop after $n$ iterations

- $y_n := x_0 \cup \ldots \cup x_n$
- $[p^*](a) := \text{"lim}_n \mu_{y_n}"$